NetLSD Documentation

Release 0.1

Anton Tsitsulin

Sep 27, 2018

Contents

Contents:

CHAPTER 1

NetLSD

NetLSD is a family of spectral graph descriptros. Given a graph, NetLSD computes a low-dimensional vector representation that can be used for different tasks.

1.1 Quick start

```
import netlsd
import networkx as nx

g = nx.erdos_renyi_graph(100, 0.01) # create a random graph with 100 nodes
descriptor = netlsd.heat(g) # compute the signature
```

That's it! Then, signatures of two graphs can be compared easily. NetLSD supports networkx, graph_tool, and igraph packages natively.

```
import netlsd
import numpy as np

distance = netlsd.compare(desc1, desc2) # compare the signatures using l2 distance
distance = np.linalg.norm(desc1 - desc2) # equivalent
```

For more advanced usage, check out online documentation.

1.2 Requirements

- numpy
- scipy

1.3 Installation

1. cd netlsd
2. pip install -r requirements.txt
3. python setup.py install

Or simply `pip install netlsd`

1.4 Citing

If you find NetLSD useful in your research, we ask that you cite the following paper:

```
@inproceedings{Tsitsulin:2018:KDD,
 author={Tsitsulin, Anton and Mottin, Davide and Karras, Panagiotis and Bronstein,␣
↪Alex and M{\"u}ller, Emmanuel},
 title={NetLSD: Hearing the Shape of a Graph},
 booktitle = {Proceedings of the 24th ACM SIGKDD International Conference on␣
↪Knowledge Discovery and Data Mining},
 series = {KDD '18},
 year = {2018},
}
```

1.5 Misc

NetLSD - Hearing the shape of graphs.

- MIT license
- Documentation: http://netlsd.readthedocs.org

CHAPTER 2

Installation

At the command line:

```
$ pip install netlsd
```

Or, if you prefer to install from source:

```
$ mkvirtualenv netlsd
$ pip install netlsd
```

CHAPTER 3

Usage

3.1 Quick start usage

```python
import netlsd
import networkx as nx

g = nx.erdos_renyi_graph(100, 0.01) # create a random graph with 100 nodes
descriptor = netlsd.heat(g) # compute NetLSD signature
```

That's it! Then, signatures of two graphs can be compared easily:

```
.. code-block:: python
```

> import netlsd
>
> distance = netlsd.compare(sig1, sig2) # compare the signatures using l2 distance

or, equivalently:

```
.. code-block:: python
```

> import numpy as np
>
> distance = np.linalg.norm(sig1 - sig2) # compare the signatures using l2 distance in numpy

3.2 Advanced usage

Here we outline different ways to get more out of NetLSD.

3.2.1 Try the wave kernel

In the paper, we introduce two kernels: heat and wave. You can simply replace `netlsd.heat` with `netlsd.wave` to switch to wave kernel. Wave kernel is known to preserve symmetries and structures as it acts as a band-pass filter on the spectrum.

3.2.2 Supply adjacency matrix directly

You do not need to use python's graph libraries to interface with NetLSD. One option is to use any type of a sparse matrix from scipy:

```
import netlsd
import scipy.sparse as sps

A = sps.random(1000, 1000) # create a random adjacency matrix
A = A + A.T # make sure it is undirected
descriptor = netlsd.heat(A) # compute NetLSD signature
```

In case you have already constructed a Laplacian, just pass it to the function.

3.2.3 Scale things up with custom eigensolvers

If you want to use a different eigensolver routine, such as SLEPc, you can directly supply eigenvalues to NetLSD:

```
import netlsd
import fancy_eigensolver

eigenvalues = fancy_eigensolver(graph)
descriptor = netlsd.heat(eigenvalues) # compute NetLSD signature
```

CHAPTER 4

Reference

4.1 NetLSD

`netlsd`	
`heat`	
`wave`	Stuff to parse WAVE files.
`compare`	

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/xgfs/NetLSD/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

5.1.4 Write Documentation

NetLSD could always use more documentation, whether as part of the official NetLSD docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/xgfs/NetLSD/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *NetLSD* for local development.

1. Fork the *NetLSD* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/NetLSD.git
   ```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv NetLSD
   $ cd NetLSD/
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 NetLSD tests
   $ python setup.py test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

CHAPTER 6

Credits

- Anton Tsitsulin <anton.tsitsulin@hpi.de>

CHAPTER 7

Indices and tables

- genindex
- modindex
- search